Architectures and Algorithms for Track Association and Fusion

Chee-Yee Chong
Booz Allen & Hamilton, Inc.
San Francisco, CA 94111

Kuo-Chu Chang
George Mason University
Dept. of System Engineering
Fairfax, VA 22030

Shozo Mori
Raytheon Systems Company
Advanced C3I Systems
San Jose, CA 95126

Abstract – Target tracking using multiple sensors can provide better performance than using a single sensor. One approach to multiple target tracking with multiple sensors is to first perform single sensor tracking and then fuse the tracks from the different sensors. Two processing architectures for track fusion are presented: sensor to sensor track fusion, and sensor to system track fusion. Technical issues related to the statistical correlation between track estimation errors are discussed. Approaches for associating the tracks and combining the track state estimates of associated tracks that account for this correlation are described and compared by both theoretical analysis and Monte Carlo simulations.

Key Words: Sensor fusion, target tracking, distributed tracking/fusion, distributed data processing

1. Introduction

The use of multiple sensors for target tracking can potentially provide better performance than a single sensor due to better visibility, complementary information, etc. Theoretically, the best tracking performance is achieved by fusing the measurements from the sensors directly. However, due to communication or organization constraints, many real-world systems have a hierarchical structure where the fusion system has no direct access to the sensor data. Instead, the sensor data are processed locally to form sensor tracks, which are then fused to form system tracks. Track fusion is then needed to associate the sensor tracks and generate an improved target state estimate.

Track fusion has technical issues that are not present in measurement fusion or centralized tracking. In particular, the state estimates of sensor tracks cannot be treated like sensor measurements and fused with a standard centralized tracking algorithm. This is due to the fact that while sensor measurement errors are usually independent across sensors and time, the errors in target state estimates associated with the tracks, i.e., tracker outputs, are generally correlated with one another. This has significant impact on the two main functions in track fusion: association and state estimate fusion. Both the computation of the association metrics and the fusion of the track state estimates need to consider any possible dependence between the track state estimation errors. The specific fusion architecture affects the nature of the statistical correlation and the algorithms that should be used.

This paper presents technical issues associated with track fusion and compares several algorithms. We first describe the track fusion problem and possible fusion architectures. This is followed by the technical issues associated with track fusion. Algorithms for track state estimate fusion and track association are then presented and compared. Specifically, we describe several algorithms for combining track state estimates, their optimality, and their advantages and disadvantages. Theoretical analysis and Monte Carlo simulations are used to compare their performance. We also describe different ways of handling the correlation between the target state estimates in computing the association metrics.

This paper focuses on deterministic target dynamics. A companion paper [1] will address the non-deterministic problem.

2. Track Fusion and Architectures

We consider a track fusion system with the components shown in Figure 1.
Single sensor tracking generates single sensor tracks and target state estimates. Periodically, the tracks from different sensors are sent to a central site to be fused.

Track fusion involves two steps: association and track state estimation fusion. In association, tracks from different sensors are associated to form system tracks, each corresponding to a single hypothesized target. Given an association, the state estimates of the system tracks are then obtained by fusing the state estimates of associated sensor tracks.

There are two possible processing architectures for track fusion depending on whether the state estimates of the system track are used.

Sensor to sensor track fusion (Figure 2). The state estimates from different sensor tracks (propagated to a common time) are associated and fused with each other to obtain the state estimate for the system track. The previous state estimate of the system track is not used in this process. Note that with this architecture, fusion will in general involve sets of tracks from more than two sensors.

This architecture does not have to deal with the problem of correlated estimation errors (if the common prior is ignored). Since it is basically a memoryless operation, the errors in association and track estimate fusion are not propagated from one time to the next. However, this approach may not be as efficient as sensor to system track fusion since past processing results are discarded.

Sensor to system track fusion (Figure 3). Whenever a set of sensor tracks is received, the state estimates of the system tracks are extrapolated to the time of the sensor tracks, and fused with the newly received sensor tracks. The process is repeated when another set of sensor tracks is received.

Sensor to system track fusion reduces the association problem to a bi-partite assignment problem so that common assignment algorithms can be used. However, it has to deal with the problem of correlated estimation errors. In Figure 3, the sensor tracks in A and the system tracks in B have correlated errors since they all depend on C. Furthermore, any errors in system tracks due to past processing errors in association or fusion will affect future fusion performance.

3. Technical Issues

The main technical issues with track fusion are due to the fact that tracks and not measurements have to be fused. The inputs to track fusion are sensor tracks formed from local measurements and represented by position and velocity estimates and their error covariance matrices.

3.1. Correlated Estimation Errors

Fusion will be relatively straightforward if the estimation errors of the two tracks to be fused are uncorrelated. The estimates can be viewed as measurements with independent errors, fused with other estimates using a standard approach (e.g., association and Kalman filter update). The estimation errors between two tracks may be correlated for the following reasons.

A. Common prior estimates. This occurs in sensor to system track fusion as in Figure 4, which shows an information graph formulation [2], [3] of the track fusion problem. The solid squares in the graph represent measurements and the hollow squares represent fusion, either of a measurement with a track or a track with another track. The tracks are assumed to have been propagated to a common time. The placement of the tracks in the graph represents information contained in the tracks. Basically a track at a node will contain all the information in the predecessor nodes (both tracks and measurements). In the example, both the sensor track estimate $\hat{x}_j$ and the system track estimate $\hat{x}_i$ contain the sensor track estimate $x_j$ propagated from an earlier time. Figure 4 also illustrates that two sensor tracks do not share common
prior estimates (except for a common prior). In general, there is correlation from this source if there are multiple paths from a measurement to the fusion node in the information graph.

B. Correlated estimation errors due to common process noise. This occurs even when fusion is between sensor tracks not sharing common measurements. The measurements from two sensor tracks are not necessarily conditionally independent given the target state at a single time when the target dynamics is not deterministic. Thus the estimation errors from two sensor tracks may not be independent.

The correlated estimation errors have to be considered in associating the tracks and in combining the state estimates for the associated tracks. Otherwise, the target state estimates in the system tracks may degrade.

Figure 4: Dependence in Track Estimates

3.2. Imperfect Association

Track fusion has to associate those sensor tracks that originate from the same targets. If the sensor tracks were pure, i.e., each sensor track consists of measurements from a single target, or the previous associations were perfect, i.e., the sensor tracks that were associated were indeed from the same targets, then track association would need only deal with new sensor tracks. At any time in the track fusion process, sensor tracks that have been previously associated should keep their previous associations. Only new sensor tracks have to be associated to determine whether they are from new targets or from previously detected targets (associated with old sensor tracks).

In practice, sensor tracks are seldom perfect. They may be impure, i.e., each sensor track may consist of measurements from different targets (misassociation), or it may be fragmented, i.e., the same target may appear in multiple sensor tracks. Furthermore, previous track associations may be incorrect in that some sensor tracks may have been mis-associated with other sensor tracks. Thus it may be necessary to re-associate sensor tracks even though they have been previously associated to system tracks. If the sensor to system track fusion processing architecture is used, the computation of the association metrics has to consider the dependence between the sensor and system tracks.

4. Track State Estimate Fusion

Track fusion consists of two steps: (1) track-to-track association, or selection of a best association hypothesis, and (2) fusion of target state estimates given an association hypothesis. We will discuss track state estimates fusion first because the same techniques can be used for computing the association metrics.

Track State Estimate Fusion problem. Suppose there are two tracks, $i$ and $j$, with state estimates and error covariance matrices (both propagated to a common time) $\hat{x}_i$ and $\hat{x}_j$, $P_i$ and $P_j$, respectively. The estimate fusion problem is to find the best fused estimate $\hat{x}$ and the error covariance matrix $P$. The two tracks may be two sensor tracks in a sensor to sensor track fusion architecture, or a system track and a sensor track in a sensor to system track fusion architecture.

Track state estimate fusion algorithms have been investigated extensively over the past two decades with most of the research performed under the topic of decentralized or distributed estimation. In the following subsections, we discuss two approaches to track fusion: “best” linear combination of track estimates and reconstruction of optimal centralized estimate.

4.1. “Best” Linear Combination of Estimates

The fused estimate is constrained to be a linear combination of the track estimates, i.e., $\hat{x} = A\hat{x}_i + B\hat{x}_j$. The matrices $A$ and $B$ are then chosen to optimize some criteria, e.g., weighted least squares or minimum variance. If the track estimates are not the sufficient statistics for the sensor measurements in the fused track, then the optimal linear combination may not be as optimal as an estimate that is allowed to use information other than the current estimates. Two algorithms have been developed for linearly combining the track estimates depending on whether the cross covariance between the track estimates is considered.

4.1.1 Basic Convex Combination

When the cross covariance between the two track estimates can be ignored, the fusion algorithm is given by [4]:

- State estimate:

$$\hat{x} = P_i (P_i + P_j)^{-1} \hat{x}_i + P_j (P_i + P_j)^{-1} \hat{x}_j$$

$$= P_i (P_i^{-1} \hat{x}_i + P_j^{-1} \hat{x}_j)$$  \hspace{1cm} (1)
• Error covariance:
\[ P = P_i - P_i(P + P_j)^{-1}P_j = P_i(P + P_j)^{-1}P_j = (P^{-1} + P_j^{-1})^{-1} \]

The basic convex combination algorithm has been used extensively because of its simple implementation. It is suboptimal when the estimation errors are correlated, such as when one track is a system track and the other track is a sensor track, or when process noise is present. However, when both tracks are sensor tracks and there is no process noise, then the fusion algorithm is (almost) optimal, i.e., it produces almost the same results as when the sensor measurements are fused directly (as will be seen in Section 4.2.4.)

4.1.2 Linear Combination with Cross Covariance

When the cross covariance between the two estimates cannot be ignored, the linear combination algorithm becomes [5]:

- State estimate:
\[ \hat{x} = \hat{x}_i + (P_i - P_j)(P_i + P_j - P_i + P_j^{-1})(\hat{x}_j - \hat{x}_i) \]  

- Error covariance:
\[ P = P_j + (P_i - P_j)(P_i + P_j - P_i + P_j^{-1})(P_j - P_i) \]  

The cross-covariance’s \( P_i \) and \( P_j \) are computed from the observation matrices and the Kalman filter gains [6].

This fusion algorithm was originally developed to account for correlation due to common process noise. However, the derivation depends only the correlation between the two estimation errors and not on the specific source of the errors, which could result from common prior estimates. The algorithm was originally thought to the optimal in the minimum mean square error (MMSE) sense but was claimed recently [7] to be only optimal in the maximum likelihood (ML) sense.

The advantage of this algorithm is its ability to handle common process noise. For example, when the estimates come from two sensor tracks, even if there is no common prior, the estimation errors may still be correlated if there is common process noise. The main disadvantage of this algorithm is the amount of information needed to compute the cross-covariance. If the system is linear and time-invariant, the cross covariance can be computed from off-line information. Otherwise, the entire history of Kalman filter gains and observation matrices need to be communicated to whoever is doing the fusion. Since extended Kalman filters are usually used for sensor level tracking, the Kalman filter gains depend on the measurement data. In this case, it may be more efficient to communicate the measurements for centralized fusion than sensor tracks for track fusion.

4.2. Reconstruction of Centralized Estimate

The other basic approach to track state estimate fusion attempts to reconstruct the optimal estimate (by fusing the measurements in the tracks) from the individual track estimates and possibly some additional information [8] – [23]. Figure 5 illustrates the philosophy behind this approach for sensor to system track fusion. The sensor measurements from the individual sensors are used to form estimates for the sensor tracks. Periodically, these estimates are fused to obtain state estimates for the system tracks. As seen in the figure, the sensor track estimate \( \hat{x}_j \) and system track estimate \( \hat{x}_i \) to be fused share the same measurements in the sensor track estimate \( \bar{x}_j \) which was transmitted earlier to fused with the system track.

The algorithms in this approach avoid double counting of information by either recognizing the common information and removing it in the fusion process or by only sending information that is uncorrelated with the system track. The latter uses the concept of so called “tracklets” [17] – [20], where a tracklet is loosely defined as a track segment computed so that its errors are not cross-correlated with the errors of other track segments.

![Figure 5: Reconstruction of Centralized Estimate](image)

4.2.1 Information De-correlation

The information de-correlation approach can be derived [9], [11], [16] easily using the information filter form of the Kalman filter. The key idea is to identify the common information in the two estimates to be fused and remove it in fusion. This approach is useful when one track is the system track and the other track is the sensor track.

The state estimate fusion algorithm is given by:

- State estimate:
\[ \hat{x} = P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j - \bar{P}_j^{-1}\bar{x}_j \]  

- Error covariance:
\[ P = (P_i^{-1} + P_j^{-1} - \bar{P}_j^{-1})^{-1} \]
where $\bar{x}_j$ and $\bar{P}_j$ are the state estimate and error covariance (propagated to the common fusion time) of the sensor track last communicated to the system track. This is the additional information that is used for the fusion algorithm. Basically, both the sensor and system tracks contain this common information. In order to avoid double counting, it has to be removed from the results.

This fusion algorithm is based upon a general theory for distributed fusion [2], [3], [16] that can support arbitrary fusion and communication architectures, e.g., fusion with feedback. The information graph introduced earlier is used to identify the common information shared by two estimates, and the fusion algorithm then avoids the double counting. In addition to fusing state estimates and their error covariances, the general approach can also be used for fusing target classification probabilities.

The main advantage of this approach is its simple implementation. No additional communication is needed since the state estimate and error covariance of the previously transmitted sensor track can be stored and propagated to the current fusion time. This approach is optimal when there is no process noise. When the process noise is small, and/or the update rate by sensor tracks is reasonably high, the degradation in performance has been shown to be small.

### 4.2.2 Equivalent Measurement

This algorithm de-correlates the sensor track by finding an equivalent measurement for the “tracklet”, i.e., the sensor measurements in the sensor track since the last communication with the system track [21]-[23].

The equivalent measurement is generated from the current estimate and error covariance and the previous estimate and error covariance of the sensor track (all propagated to a common time) as follows:

- **Equivalent measurement**
  \[
  u_j = \bar{x}_j + \bar{P}_j (\bar{P}_j - P_j)^{-1} (\hat{x}_j - \bar{x}_j) \tag{7}
  \]

- **Error covariance**
  \[
  U_j = \bar{P}_j (\bar{P}_j - P_j)^{-1} P_j \tag{8}
  \]

The error of the equivalent measurement is conditionally uncorrelated with the estimation error of the global track. Thus, the standard Kalman update equation can be used to combine the equivalent measurement with the current state estimate. More specifically, the update equation is:

- **State estimate:**
  \[
  \hat{x} = \hat{x}_j + \bar{P}_j (\bar{P}_j + U_j)^{-1} (u_j - \hat{x}_j) \tag{9}
  \]

- **Error covariance:**
  \[
  P = \bar{P}_j - \bar{P}_j (\bar{P}_j + U_j)^{-1} \bar{P}_j \tag{10}
  \]

Equations (9) and (10) with (7) and (8) are completely equivalent to (5) and (6) in Section 4.2.1. Equations (5) and (6) are in the information matrix form while (7) – (10) are in the covariance matrix form. Hence, numerical calculation issues aside, the performance and behavior of the two algorithms should be exactly the same. The information de-correlation algorithm has the advantage that the same approach can be used for general non-Gaussian or discrete probability distributions.

### 4.2.3 Restarting Local Filters

Another way of de-correlating the sensor track from the system track is to generate local track estimates using only the measurements since the last communication. This approach is sometimes called “tracklets from measurements” [18]. As seen in Figure 6, the local and system tracks are de-correlated since they do not share any common information. In terms of the information graph, there is a single unique path from each measurement to the fusion node.

![Figure 6: Restarting Local Filters](image-url)

In this approach, after a sensor track has been transmitted to be fused with the system track, the local filter is re-started using the new measurements. The estimate from these measurements is then uncorrelated with the system track and can be fused easily with the system track. Since sensor tracking needs good estimates to evaluate the matrices for the extended Kalman filter and to support association, the usual tracker that processes all measurements is also used. Thus the local filter that is restarted periodically can be viewed as the “shadow tracker” [19].

The advantage of this approach is its simplicity. The disadvantage is the need to modify the existing tracking algorithm for the sensors. This approach is also equivalent to Equations (5) – (10).

### 4.2.4 Global Restart

The main problem in track fusion is the correlation between the sensor and system tracks. The correlation problem does not exist if the state estimate of
the system track is not used in fusing the sensor track estimates. Since the sensor tracks already contain all the available measurements up to the current time, this global estimate formed will be optimal (Figure 7).

At each fusion time, the estimates of the sensor tracks are fused with each other to obtain the global estimate. The fusion algorithm is given by:

- **State estimate:**
  \[
  \hat{x} = P(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j - \overline{P}^{-1}\overline{x})
  \]  
  (11)

- **Error Covariance:**
  \[
  P = (P_i^{-1} + P_j^{-1} - \overline{P}^{-1})^{-1}
  \]  
  (12)

where \(\overline{x}\) and \(\overline{P}\) are the common prior state and covariance used by the sensor trackers propagated to the current time. Note that even though (11) and (12) are the same as (5) and (6), they are based on different processing architectures and use different priors.

When the prior covariance matrix \(\overline{P}\) is much larger than the updated estimation error covariance matrix, or when it becomes very large due to forward propagation, then \(\overline{P}^{-1} \rightarrow 0\), and these equations are equivalent to the basic convex combination equations (1) and (2) described before, i.e.,

\[
\hat{x} = P(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j)
\]  
(13)

\[
P = (P_i^{-1} + P_j^{-1})^{-1}
\]  
(14)

![Figure 7: Global Restart](image)

The advantage of this approach is that it does not have to perform any de-correlation since the sensor tracks to be fused do not have correlated estimation errors.

4.3. Numerical Results

We present some numerical results to compare the performance of the fusion algorithms.

4.3.1 Simulation Approach

Two sensors were located such that there is overlapping coverage of the target trajectory and the viewing geometry offers potential performance improvement from fusion. Measurements from the two sensors were then simulated.

Single sensor tracking algorithms were simulated by standard extended Kalman filters. The outputs from the two single sensor trackers were periodically fused using the algorithms to be analyzed.

The estimated target states (position and velocity) were compared with the true target states to find the estimation error. The errors from multiple Monte Carlo runs were averaged to find the root mean square (RMS) position and velocity errors. The parameter to be varied is the communication or fusion frequency.

We compared the performance of two fusion algorithms representing the two main approaches – convex combination and tracklet by means of information de-correlation. The performance of single sensor tracking and centralized tracking, was computed analytically by the Cramer Rao bounds. The two reference cases provide lower and upper bounds for what can be achieved with the sensor measurements.

The Cramer Rao bound provides a theoretical lower bound on the estimation error covariance matrix that is achievable by an unbiased nonlinear estimate [24]. For continuous time nonlinear deterministic models with discrete-time nonlinear measurements with additive Gaussian white noise, it can be shown [25] that the extended Kalman filter covariance propagation equations linearized about the true (unknown) trajectory provide the Cramer Rao bound to the estimation error covariance matrix. This is computationally far more efficient than Monte Carlo runs.

4.3.2 Simulation Results

Figures 8 to 11 show the Monte Carlo performance results of the convex combination and tracklet fusion algorithms for 10 sec. communication interval and the Cramer Rao bounds for single sensors and both sensors (centralized fusion). The local sensor observation interval is 1 sec. Note that there is definitely benefit in fusion. Both fusion algorithms essentially achieve the performance of centralized fusion (measurement fusion) as predicted by the Cramer Rao bounds. This is consistent with the theoretic analysis that shows the optimality of both algorithms. The results also show that the performance right after communication does not seem to be affected by the communication frequencies simulated. Nominal values in the figures are 1 Km. for position and 1 m/s for velocity.
5. Track Association

Before the track state estimates can be combined, the sensor tracks have to be associated either with each other (sensor track to sensor track fusion) or with the system tracks (sensor track to system track fusion). As discussed before, if the sensor tracks were perfect and the previous associations can be trusted, then only the new sensor tracks need to be associated.

The state estimates of the other sensor tracks are fused with the system tracks that contain them to update the estimates. In practice, both the sensor tracks and the previous associations may have errors. Thus some form of re-association is needed. In a multiple hypothesis approach, this is handled naturally as the probabilities of the association hypotheses are re-evaluated. In a single hypothesis approach, the system tracks and sensor tracks are evaluated to determine which ones need to be re-associated along with the new sensor tracks.

Track association consists of the two key steps: computing a table of association metrics and selecting the best association hypothesis, usually by some assignment algorithm.

5.1. Track Association Metrics

The association metric measures how close one track is to another so that association decisions can be made. A traditional association metric is the squared Mahalanobis distance. Given two tracks \( i \) and \( j \) with mean estimates and covariances represented by \((\hat{x}_i, V_i)\) and \((\hat{x}_j, V_j)\), the Mahalanobis distance is defined as:

\[
\chi_{ij} = \sqrt{(\hat{x}_i - \hat{x}_j)^T (V_i + V_j)^{-1} (\hat{x}_i - \hat{x}_j)}
\]

The association metric has to be modified when the track state estimation errors are correlated. The problem was considered by Bar-Shalom [6] when the correlation was due to common process noise and the result is similar to that in (3) and (4). Even when process noise is absent, one still has to be careful. In particular, in computing the association metric between a sensor track and a system track that contains the sensor track, one cannot ignore the correlation between these tracks. In the following we discuss how the association metrics should be modified depending on the quality of the sensor track or quality of the previous association.

5.1.1 Imperfect Association

When the previous association between a sensor track and a system track is questionable, the sensor track and the system track need to be re-associated with other system tracks and sensor tracks. Since the sensor track and system track have correlated estimation errors, the association metric has to account for the correlation in order to give correct results. In Figure 12, the system track shown was previously associated with Sensor Track 1 and Sensor Track 2 (and
possibly other sensor tracks). Suppose it is necessary to reconsider the association between the system track and the sensor track due to large Mahalanobis distance between the tracks. Assume also that the purity of the sensor track 1 can be trusted. The effect of the sensor track 1 has to be removed from the system track before the association metric can be computed. Otherwise the system track will be too close to the sensor track. The error covariance and state estimate of the de-correlated system track is given by:

\[
(V_{\text{sys}})^{-1} = V_i^{-1} - V_j^{-1}
\]

\[
(V_{\text{sys}})^{-1} \hat{x}_{\text{sys}} = V_i^{-1} \hat{x}_i - V_j^{-1} \hat{x}_j
\]

Essentially we have replaced the sensor track 1 with the tracklet since the last association with the system track (Figure 13). This de-correlated sensor track is then used to compute the association metric with the system track.

6. Summary

In this paper we have considered the track fusion problem and technical issues including correlated estimation errors in the tracks, imperfect sensor tracks and impure previous association. Among these, correlated estimation errors depend on the fusion processing architecture and affect the choice of track fusion algorithms. We presented different approaches for fusing track state estimates, and compared their performance through theoretical analysis and Monte Carlo simulations. We also discussed different approaches for computing the association metric.

7. References


